链滴

SpringBoot 系列 -- 日志配置

作者：Qiyue0726
原文链接：https://ld246.com/article/1564744709538
来源网站：链滴

导入 Maven 包

SpringBoot 默认使用了 LogBack 日志系统，所以如无需求是没有必要改为其他日志系统的，也无再进行多余的配置，LogBack 默认将日志打到控制台上。

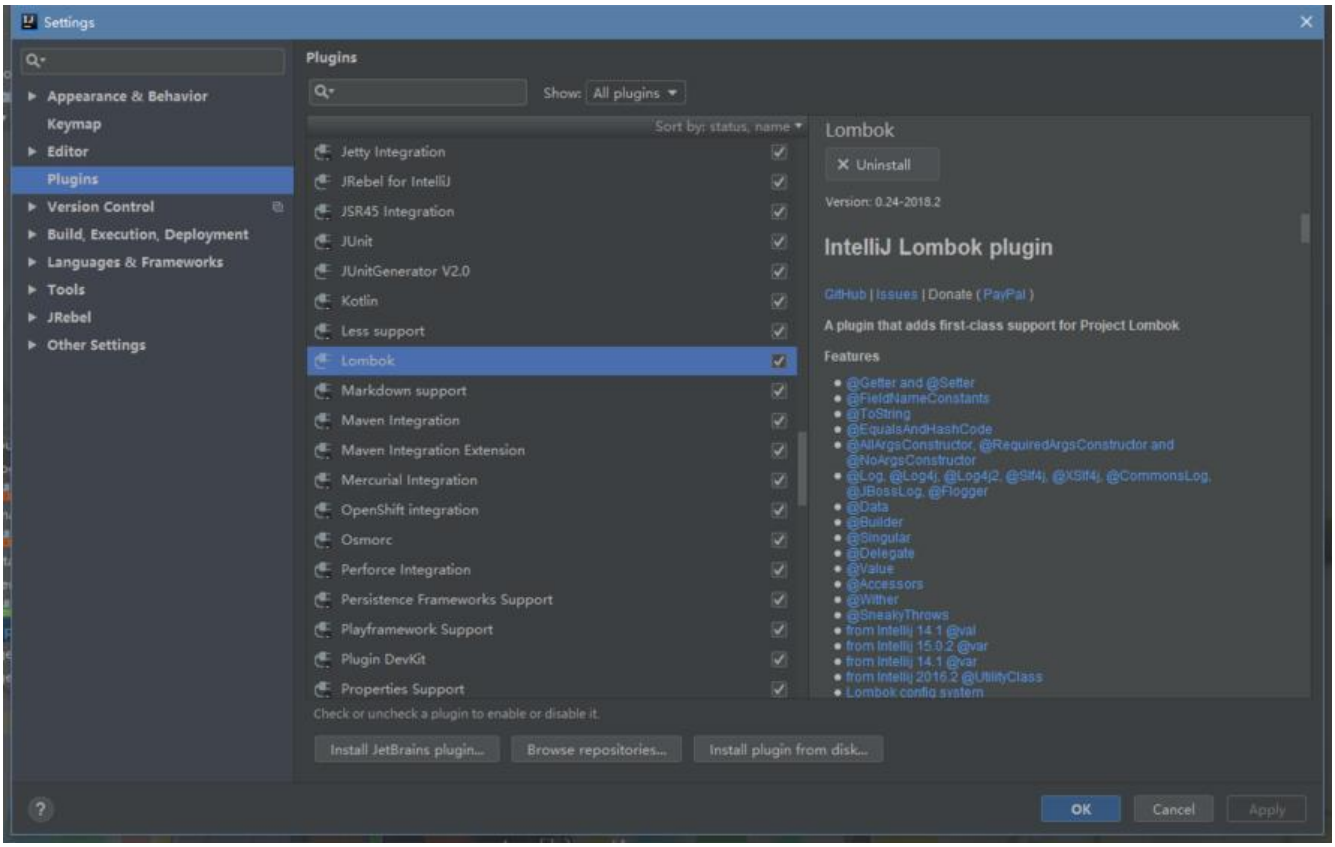
当我们新建 SpringBoot 项目时，Maven 会自动引用 spring-boot-starter 或 spring-boot-starter-eb，而这两个起步依赖中已经包含了对于 spring-boot-starter-logging 的依赖，所以无需再额外添日志系统依赖，开箱即用。

但在这里，为了方便使用日志，我们将导入 Slf4j 和 Lombok

```
<!--@Slf4j需导入lombok-->
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>1.7.26</version>
</dependency>

<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.8</version>
    <scope>provided</scope>
</dependency>
```

注： 要使用 @Slf4j 注解需要导入 Lombok 插件，而使用 Lombok 插件需要 IDEA 安装该插件。



配置

1. **在 resource 文件夹下新建 log 文件夹，并在其下新建 logback.xml 文件。**

```
<!-- Logback configuration. See http://logback.qos.ch/manual/index.html -->

<configuration scan="true" scanPeriod="10 seconds">
    <!--继承spring boot提供的logback配置-->
    <!--<include resource="org/springframework/boot/logging/logback/base.xml" />-->

    <!--设置系统日志目录-->
    <property name="APP_DIR" value="spring-boot-log"/>

    <!-- 彩色日志 -->
    <!-- 彩色日志依赖的渲染类 -->
    <conversionRule conversionWord="clr" converterClass="org.springframework.boot.logging
logback.ColorConverter"/>
    <conversionRule conversionWord="wex" converterClass="org.springframework.boot.loggi
g.logback.WhitespaceThrowableProxyConverter"/>
    <conversionRule conversionWord="wEx" converterClass="org.springframework.boot.loggi
g.logback.ExtendedWhitespaceThrowableProxyConverter"/>
    <!-- 彩色日志格式 -->
    <property name="CONSOLE_LOG_PATTERN"
              value="${CONSOLE_LOG_PATTERN:-%clr(%d{yyyy-MM-dd HH:mm:ss.SSS}){faint} %clr
${LOG_LEVEL_PATTERN:-%5p}) %clr(${PID:- }){magenta} %clr(---){faint} %clr([%15.15t]){faint}
clr(%-40.40logger{39}){cyan} %clr(:){faint} %m%n${LOG_EXCEPTION_CONVERSION_WORD:-%
Ex}}"/>

    <!-- 控制台输出 -->
```

```
    <appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <Pattern>${CONSOLE_LOG_PATTERN}</Pattern>
            <charset>UTF-8</charset> <!-- 此处设置字符集 -->
        </encoder>
        <!--此日志appender是为开发使用，只配置最底级别，控制台输出的日志级别是大于或等于此
别的日志信息-->
        <filter class="ch.qos.logback.classic.filter.ThresholdFilter">
            <level>info</level>
        </filter>
    </appender>

    <!-- 时间滚动输出 level为 DEBUG 日志 -->
    <appender name="DEBUG_FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <!-- 正在记录的日志文件的路径及文件名 -->
        <file>${LOG_PATH}/log_debug.log</file>
        <!--日志文件输出格式-->
        <encoder>
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %msg%
</pattern>
            <charset>UTF-8</charset> <!-- 此处设置字符集 -->
        </encoder>
        <!-- 日志记录器的滚动策略，按日期，按大小记录 -->
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <!--
                归档的日志文件的路径，例如今天是2017-04-26日志，当前写的日志文件路径为file节点
定，可以将此文件与file指定文件路径设置为不同路径，从而将当前日志文件或归档日志文件置不同
目录。
                而2017-04-26的日志文件在由fileNamePattern指定。%d{yyyy-MM-dd}指定日期格式，
i指定索引
            -->
            <fileNamePattern>${LOG_PATH}/debug/log-debug-%d{yyyy-MM-dd}.%i.log</fileNa
ePattern>
            <!--
                除按日志记录之外，还配置了日志文件不能超过500M，若超过500M，日志文件会以索引
开始，
                命名日志文件，例如log-error-2017-04-26.0.log
            -->
            <timeBasedFileNamingAndTriggeringPolicy class="ch.qos.logback.core.rolling.SizeAn
TimeBasedFNATP">
                <maxFileSize>500MB</maxFileSize>
            </timeBasedFileNamingAndTriggeringPolicy>
            <!--日志文件保留天数-->
            <maxHistory>30</maxHistory>
        </rollingPolicy>
        <!-- 此日志文件只记录debug级别的 -->
        <filter class="ch.qos.logback.classic.filter.LevelFilter">
            <level>debug</level>
            <onMatch>ACCEPT</onMatch>
            <onMismatch>DENY</onMismatch>
        </filter>
    </appender>

    <!-- 时间滚动输出 level为 INFO 日志 -->
```

```
    <appender name="INFO_FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <!-- 正在记录的日志文件的路径及文件名 -->
        <file>${LOG_PATH}/log_info.log</file>
        <!--日志文件输出格式-->
        <encoder>
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %msg%
</pattern>
            <charset>UTF-8</charset> <!-- 此处设置字符集 -->
        </encoder>
        <!-- 日志记录器的滚动策略，按日期，按大小记录 -->
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <!--
                归档的日志文件的路径，例如今天是2017-04-26日志，当前写的日志文件路径为file节点
定，可以将此文件与file指定文件路径设置为不同路径，从而将当前日志文件或归档日志文件置不同
目录。
                而2017-04-26的日志文件在由fileNamePattern指定。%d{yyyy-MM-dd}指定日期格式，
i指定索引
            -->
            <fileNamePattern>${LOG_PATH}/info/log-info-%d{yyyy-MM-dd}.%i.log</fileNamePat
ern>
            <!--
                除按日志记录之外，还配置了日志文件不能超过500M，若超过500M，日志文件会以索引
开始，
                命名日志文件，例如log-error-2017-04-26.0.log
            -->
            <timeBasedFileNamingAndTriggeringPolicy class="ch.qos.logback.core.rolling.SizeAn
TimeBasedFNATP">
                <maxFileSize>500MB</maxFileSize>
            </timeBasedFileNamingAndTriggeringPolicy>
            <!--日志文件保留天数-->
            <maxHistory>30</maxHistory>
        </rollingPolicy>
        <!-- 此日志文件只记录info级别的 -->
        <filter class="ch.qos.logback.classic.filter.LevelFilter">
            <level>info</level>
            <onMatch>ACCEPT</onMatch>
            <onMismatch>DENY</onMismatch>
        </filter>
    </appender>

    <!-- 时间滚动输出 level为 WARN 日志 -->
    <appender name="WARN_FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <!-- 正在记录的日志文件的路径及文件名 -->
        <file>${LOG_PATH}/log_warn.log</file>
        <!--日志文件输出格式-->
        <encoder>
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %msg%
</pattern>
            <charset>UTF-8</charset> <!-- 此处设置字符集 -->
        </encoder>
        <!-- 日志记录器的滚动策略，按日期，按大小记录 -->
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <!--
                归档的日志文件的路径，例如今天是2017-04-26日志，当前写的日志文件路径为file节点
```

```
定，可以将此文件与file指定文件路径设置为不同路径，从而将当前日志文件或归档日志文件置不同
目录。
            而2017-04-26的日志文件在由fileNamePattern指定。%d{yyyy-MM-dd}指定日期格式，
i指定索引
        -->
        <fileNamePattern>${LOG_PATH}/warn/log-warn-%d{yyyy-MM-dd}.%i.log</fileName
attern>
        <!--
            除按日志记录之外，还配置了日志文件不能超过500M，若超过500M，日志文件会以索引
开始，
            命名日志文件，例如log-error-2017-04-26.0.log
        -->
        <timeBasedFileNamingAndTriggeringPolicy class="ch.qos.logback.core.rolling.SizeAn
TimeBasedFNATP">
            <maxFileSize>500MB</maxFileSize>
        </timeBasedFileNamingAndTriggeringPolicy>
        <!--日志文件保留天数-->
        <maxHistory>30</maxHistory>
    </rollingPolicy>
    <!-- 此日志文件只记录warn级别的 -->
    <filter class="ch.qos.logback.classic.filter.LevelFilter">
        <level>warn</level>
        <onMatch>ACCEPT</onMatch>
        <onMismatch>DENY</onMismatch>
    </filter>
</appender>

<!-- 时间滚动输出 level为 ERROR 日志 -->
<appender name="ERROR_FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
    <!-- 正在记录的日志文件的路径及文件名 -->
    <file>${LOG_PATH}/log_error.log</file>
    <!--日志文件输出格式-->
    <encoder>
        <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %msg%
</pattern>
        <charset>UTF-8</charset> <!-- 此处设置字符集 -->
    </encoder>
    <!-- 日志记录器的滚动策略，按日期，按大小记录 -->
    <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
        <!--
            归档的日志文件的路径，例如今天是2017-04-26日志，当前写的日志文件路径为file节点
定，可以将此文件与file指定文件路径设置为不同路径，从而将当前日志文件或归档日志文件置不同
目录。
            而2017-04-26的日志文件在由fileNamePattern指定。%d{yyyy-MM-dd}指定日期格式，
i指定索引
        -->
        <fileNamePattern>${LOG_PATH}/error/log-error-%d{yyyy-MM-dd}.%i.log</fileName
attern>
        <!--
            除按日志记录之外，还配置了日志文件不能超过500M，若超过500M，日志文件会以索引
开始，
            命名日志文件，例如log-error-2017-04-26.0.log
        -->
        <timeBasedFileNamingAndTriggeringPolicy class="ch.qos.logback.core.rolling.SizeAn
```

```
TimeBasedFNATP">
            <maxFileSize>500MB</maxFileSize>
        </timeBasedFileNamingAndTriggeringPolicy>
        <!--日志文件保留天数-->
        <maxHistory>30</maxHistory>
    </rollingPolicy>
    <!-- 此日志文件只记录ERROR级别的 -->
    <filter class="ch.qos.logback.classic.filter.LevelFilter">
        <level>error</level>
        <onMatch>ACCEPT</onMatch>
        <onMismatch>DENY</onMismatch>
    </filter>
</appender>

<logger name="org.springframework.web" level="info"/>
<logger name="org.springframework.scheduling.annotation.ScheduledAnnotationBeanPos
Processor" level="INFO"/>
<!--name为当前工程的java代码的完整目录-->
<logger name="com.sakura.anima" level="debug"/>

<!--开发环境:打印控制台-->
<springProfile name="dev">
    <root level="info">
        <appender-ref ref="CONSOLE"/>
        <appender-ref ref="DEBUG_FILE"/>
        <appender-ref ref="INFO_FILE"/>
        <appender-ref ref="WARN_FILE"/>
        <appender-ref ref="ERROR_FILE"/>
    </root>
</springProfile>

<!--测试环境:打印控制台和输出到文件-->
<springProfile name="test">
    <root level="info">
        <appender-ref ref="CONSOLE"/>
        <appender-ref ref="INFO_FILE"/>
        <appender-ref ref="WARN_FILE"/>
        <appender-ref ref="ERROR_FILE"/>
    </root>
</springProfile>

<!--生产环境:输出到文件-->
<springProfile name="prod">
    <root level="error">
        <appender-ref ref="CONSOLE"/>
        <appender-ref ref="DEBUG_FILE"/>
        <appender-ref ref="INFO_FILE"/>
        <appender-ref ref="ERROR_FILE"/>
    </root>
</springProfile>

</configuration>
```

2. **配置 application.yml**

```
#日志
logging:
  #日志配置文件位置
  config: classpath:log/logback.xml
  #日志打印位置，这里是默认在项目根路径下
  path: log/spring-boot-log
```

注：

以上是比较详尽的配置，如果不需要，可直接在 application.xml 下进行如下配置，此时可无需配置 lgback.xml

```
#日志
logging:
  #日志打印位置，这里是默认在项目根路径下
  path: log/spring-boot-log

  #日志文件名，默认为spring.log，和logging.path不同时生效
  file: mylog.log
  #亦可配置指定的日志输出级别
  level:
    root: info
```

开始使用

```
@RestController
@Slf4j
public class IndexController {

    @RequestMapping(value = "/")
    public String index(){

        log.info("使用日志");

        return "hello world";
    }

}
```

```
2019-08-01 16:18:59.269  INFO 7804 --- [           main] com.sakura.anima.AnimaApplication        : Started AnimaApplication in 5.625 secon
2019-08-01 16:19:10.453  INFO 7804 --- [p-nio-80-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet '
2019-08-01 16:19:10.453  INFO 7804 --- [p-nio-80-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet
2019-08-01 16:19:10.468  INFO 7804 --- [p-nio-80-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 14 ms
2019-08-01 16:19:10.505  INFO 7804 --- [p-nio-80-exec-1] c.s.anima.controller.IndexController     : 使用日志
```

如上，在类上添加 @Slf4j 注解就可以直接使用日志